REMARKS

Initially, Applicants thank the Examiner for the courtesies extended during the recent in-person interview held on August 14. The claim amendments and arguments submitted in this paper are consistent with the amendments and arguments presented during the course of the interview.

The Office Action mailed June 21, 2007, considered and rejected claims 1-7, 9-16, 18-26, 28-33, 35-42, 45 and 46. Claims 1-7, 9-16, 18-26, 28-33, 35-42, 45 and 46 were rejected under 35 U.S.C. § 103(a) as being unpatentable over *Barnes et al.* (U.S. Patent No. 7,096,419) in view of *Chinnici et al.* (U.S. Pub. No 2003/0191803) and in view of *Ardoin et al.* (U.S. Patent No. 6,052,691).[1]

By this amendment claims 1, 2, 10, 11, 13, 19, 20, 25, 28, 29, 30, 35, 36 and 41 have been amended.[2] Claims 1-7, 9-16, 18-26, 28-33, 35-42, 45 and 46 are pending, of which claims 1, 11, 20, 29 and 36 are the only independent claims at issue.

The present invention is generally directed to serializing an object from an initial representation to any of one or more subsequent representations. For example, claim 1 defines providing a serialization manager to (i) coordinate one or more standard serialization providers that each identify one or more standard serializers for a standard object type or serialization format, and (ii) load, as needed, one or more custom serialization providers that each identify one or more custom serializers for one or more custom object types or serialization formats that are not covered by the one or more standard serialization providers. Next, claim 1 defines requesting a first serializer from the serialization manager for an object graph that comprises a root object and one or more other objects of particular object types, and for a particular serialization format, wherein the first serializer is requested from the serialization manager as part of a cut, copy or paste operation on the object graph, the first serializer serializing the root object to produce a new snippet of code sufficient to undo or redo a change in the object graph, the first serializer producing the snippet of code to include a new type definition.

---

[1] Although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

[2] Support for the amendments to the claims are found throughout the specification and previously presented claims, including but not limited to paragraphs [0035]-[0039] and Figures 1 & 3.

Claim 1 further defines requesting at least one other serializer from the serialization manger for the one or more other objects of the particular object types, and for a respective particular serialization format, wherein the at least one other serializer is requested from the serialization manager as part of the cut, copy or paste operation on the object graph, and such that due to the cut, copy or paste operation, the at least one other serializer produces an additional snippet of code sufficient to undo or redo a change to the object graph, the production of the code snippet preventing production of a new class representation of the one or more other objects of the object graph, such that instead of producing a new class representation comprising at least some portion of source code, a visual representation of the object graph is generated. Lastly, claim 1 defines calling the serializer to serialize the object graph.

Claims 11, 20, 29 and 36 are directed to methods and computer program products and generally correspond to the method of claim 1. Applicants respectfully submit that the cited art of record does not anticipate or otherwise render the amended claims unpatentable for at least the reason that the cited art does not disclose, suggest, or enable each and every element of these claims.

Barnes generally relates to a system for serializing JavaBean state information into XML trees and structures within an application builder (Col. 2:34-54). A frame development includes a frame that houses JavaBeans in order to provide a GUI for display and manipulation by a user (Col. 3:23-25). As the display is manipulated, a node tree generation module is used to generate an object state node tree that captures state information of objects used within the frame (Col. 3:32-34). A file is created that contains the object state information of the JavaBean in an XML tagged format that fully specifies the public and private states of the objects in the frame (Col. 3:62-63).

Chinnici generally relates to deserialization of objects and, in particular, of supporting extensible type mappings between XML data types and Java types, in which there is a Java type serializer and an XML type deserializer (¶¶ 126, 128). The extensible type mappings may be used by an API to allow the format of data to be changed from XML to Java based on the type mapping defined between the corresponding Java type and the XML data type, as it is moved from a client to server (¶¶ 126-129). For instance, a serializer interface may include a serializer for Java arrays and classes and use an XML schema definition to generate a corresponding XML representation from the Java object (¶ 129).

Ardoin relates to a method for maintaining relationships between entities in a computer system, and particularly to CAD/CAM software systems for managing objects so that when an object is modified, copied, or deleted, referential integrity is maintained, thereby allowing undoing or reversing actions taken on the object (Col. 1, ll. 32-43; Col. 3, ll. 13-29). More particularly, *Ardoin* describes a system which includes a relations subsystem that describes the associative and constrain relationships between various objects by using an associative graph (Col. 6, ll. 28-35). Within the associative graph are nodes that are connected and represent related entities, and edges between the nodes, which represent the relationships between the entities (Col. 6, ll. 28-44). The graph itself may be modified when nodes/entities are modified, copied, or deleted (Col. 6, ll. 35-39). Further, the system of Ardoin can include an optional undo interface. When objects are modified, the undo interface writes modifications to a log, which log can thereafter be read to restore a value (Col. 41, ll. 47-54).

None of the cited art, however, describes or suggests requesting a first serializer from the serialization manager for an object graph that comprises a root object and one or more other objects of particular object types, and for a particular serialization format, wherein the first serializer is requested from the serialization manager as part of a cut, copy or paste operation on the object graph, the first serializer serializing the root object to produce a new snippet of code sufficient to undo or redo a change in the object graph, the first serializer producing the snippet of code to include a new type definition, as recited in claim 1.

Furthermore, none of the cited art teaches or suggests requesting at least one other serializer from the serialization manger for the one or more other objects of the particular object types, and for a respective particular serialization format, wherein the at least one other serializer is requested from the serialization manager as part of the cut, copy or paste operation on the object graph, and such that due to the cut, copy or paste operation, the at least one other serializer produces an additional snippet of code sufficient to undo or redo a change to the object graph, the production of the code snippet preventing production of a new class representation of the one or more other objects of the object graph, such that instead of producing a new class representation comprising at least some portion of source code, a visual representation of the object graph is generated, as recited in claim 1.

At least for either of these reasons, claim 1 patentably defines over the art of record. At least for either of these reasons, claims 11, 20, 29 and 36 also patentably define over the art of

record. Since each of the dependent claims depend from one of claims 1, 11, 20, 29 and 36, each of the dependent claims also patentably define over the art of record for at least either of the same reasons.

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice. Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting the teachings officially noticed, as well as the required motivation or suggestion to combine the relied upon notice with the other art of record.

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at (801) 533-9800.

Dated this 21st day of September, 2007.

Respectfully submitted,

/GREGORY R. LUNT/

RICK D. NYDEGGER
Registration No. 28,651
JENS C. JENKINS
Registration No. 44,803
COLBY C. NUTTALL
Registration No. 58,146
GREGORY R. LUNT
Registration No. 57,354
Attorneys for Applicant
Customer No. 047973

RDN:JCJ:CCN:GRLgd
GD0000002108V001